On Formalising Dialogue Systems for Argumentation in Event Calculus

Lianne Bodenstaff
Department of Computer Science
University of Twente
The Netherlands
l.bodenstaff@ewi.utwente.nl

Henry Prakken
Department of Information and
Computing Sciences
Utrecht University
and
Faculty of Law
University of Groningen
The Netherlands
henry@cs.uu.nl

Gerard Vreeswijk
Department of Information and
Computing Sciences
Utrecht University
The Netherlands
gv@cs.uu.nl

Abstract

This paper studies the logical formalisation and implementation of dialogue systems for argumentation, motivated by the claim that this benefits their formal investigation and implementation. A case study is described in which a dialogue system of Prakken is formalised in Shanahan's version of the 'full' Event Calculus and then implemented as a Prolog program. Then a second case study is briefly summarised in which a dialogue system of Parsons, Wooldridge and Amgoud is formalised in the same way. From the case studies some conclusions are drawn on the usefulness of the formalisation method.

Introduction

Logical specification of dialogue systems benefits both the formal investigation of such systems and their implementation in declarative programming languages. Such implementation in turn supports the design of flexible dialogue systems, in which variations in the communication language or protocol can be handled easier than when they are hard-coded in a lower-level programming language. This paper studies the logical specification of a class of dialogue systems that have so far largely been specified in semi-formal ways, viz. systems involving argumentation.

When intelligent agents interact, the need for argumentation can arise in various ways. For instance, collaborating agents who must jointly solve a problem may argue about the pros and cons of the various possible solutions (Atkinson, Bench-Capon, & McBurney 2005), or self-interested negotiating agents may try to persuade each other to accept their offers by arguing about the merits and drawbacks of these offers (Rahwan *et al.* 2003). Various dialogue systems for persuasive argumentation have been proposed, e.g. (Gordon 1994; Amgoud, Maudet, & Parsons 2000; Parsons, Wooldridge, & Amgoud 2003; Atkinson, Bench-Capon, & McBurney 2005; Prakken 2005) but most of them have so far not been fully formally specified in a declarative way. Two notable exceptions are Brewka (2001) and Artikis, Sergot, & Pitt (2003). However, these papers study a single and particular system that lacks several features of other systems, which therefore still await further investigation.

Locution	Description	P	PWA
claim φ	a player claims that φ is the case	x	x
claim S	a player claims the set of propositions *S*		x
why φ	a player asks why φ is the case	x	x
concede φ	a player concedes proposition φ	x	x
concede S	a player concedes the set of propositions *S*		x
argue A	a player puts forward argument *A*	x	
retract φ	a player retracts proposition φ	x	

Table 1: Some speech acts for argumentation

This paper aims at contributing to such investigation by carrying out a case study in which a dialogue system for argumentation of Prakken (2005) is formalised in a variant of the Event Calculus and then implemented as a Prolog program. After that a second case study will be briefly summarised in which a system of Parsons, Wooldridge, & Amgoud (2003) was formalised in the same way. We end with a discussion of related research and some conclusions.

The system to be formalised

Like most dialogue systems for argumentation, the one of Prakken (2005) is specified as a dialogue game. Such a game firstly has a *topic language* L_t with a logic $\mathcal{L}$, specifying how to represent and reason with domain information. Secondly, a dialogue game has a *communication language* L_c with a *protocol* P: the former specifies the well-formed locutions, or speech acts, while the latter regulates their use. Table 1 displays a possible set of locutions for argumentation dialogues and indicates which of them are part of Prakken (2005) (P) and Parsons, Wooldridge, & Amgoud (2003) (PWA). In this table, the locution $argue\ A$ consists of premises ($prem(A)$) and conclusion ($conc(A)$). The protocol of a dialogue game specifies the 'rules of the game', i.e., it specifies the allowed moves at each point in a dialogue. A dialogue game also has *effect rules*, which specify the effects of utterances on the players' commitments. For instance, an utterance of *claim* φ initiates the speaker's commitment to φ while the utterance of *retract* φ terminates such commitment. Finally, a dialogue game has *termination rules* and sometimes *outcome rules*.

We now specify the P system in more detail. In fact, the system specified below is an instance of a framework allowing for alternative instantiations on the just-mentioned building blocks. The protocol of P is very liberal in its structural aspects: essentially, both players can speak whenever they like, except that they cannot speak at the same time. Also, they may reply to any earlier move of the other player instead of having to reply to the last such move, and they may move alternative replies to the same move, possibly even in the same turn (a turn is a sequence of moves of one player). Other protocols defined in (Prakken 2005) impose restrictions on these points; one of them, concerning relevance of moves, will be briefly discussed later below.

The **topic language** and its **logic** are assumed to be some argumentation logic fitting the format of Dung's (1995) in which arguments can be formed by chaining deductive and defeasible inference rules into trees and in which arguments can be defeated on their use of defeasible inference rules. Dialogues are between a *proponent* P and *opponent* O of a single *dialogue topic* $t \in L_t$. The protocol is based on the following ideas. Each dialogue move except the initial one replies to one earlier move in the dialogue of the other party (its *target*). Thus a dialogue can be regarded in two ways: as a sequence (reflecting the order in which the moves are made) and as a tree (reflecting the reply relations between the moves). Each replying move is either an *attacker* or a *surrender*. For instance, a *claim* p move can be attacked with a *why* p move and surrendered with a *concede* p move; and a *why* p move can be attacked with an *argue* A move where A is an argument with conclusion p, and surrendered with a *retract* p move. When s is a surrendering and s' is an attacking reply to s'', we say that s' is an *attacking counterpart* of s. The **communication language** L_c is specified in Table 2. In this table, φ is from L_t and arguments A and B are well-formed arguments from $\mathcal{L}$, while defeat relations between arguments are determined according to $\mathcal{L}$.

Acts	Attacks	Surrenders
claim φ	*why* φ	*concede* φ
why φ	*argue* A ($conc(A) = \varphi$)	*retract* φ
argue A	*why* φ ($\varphi \in prem(A)$), *argue* B (B defeats A)	*concede* φ ($\varphi \in prem(A)$ or $\varphi = conc(A)$)
concede φ		
retract φ		

Table 2: Reply structure

The protocol for L_c is defined in terms of the notion of a **dialogue**, which in turn is defined with the notion of a **move**.

- The set M of *moves* is defined as $\mathbb{N} \times \{P, O\} \times L_c \times \mathbb{N}$, where the four elements of a move m are denoted by, respectively:
  - $id(m)$, the *identifier* of the move,
  - $pl(m)$, the *player* of the move,
  - $s(m)$, the *speech act* performed in the move,
  - $t(m)$, the *target* of the move.
- The set of finite *dialogues*, denoted by $M^{<\infty}$, is the set of all finite sequences $m_1, \ldots, m_i, \ldots$ from M such that
  - each i^{th} element in the sequence has identifier i,
  - $t(m_1) = 0$;
  - for all $i > 1$ it holds that $t(m_i) = j$ for some m_j preceding m_i in the sequence.

  For any dialogue $d = m_1, \ldots, m_n, \ldots$, the sequence $m_1, \ldots, m_i$ is denoted by d_i, where d_0 denotes the empty dialogue.

When $t(m) = id(m')$ we say that m replies to m' in d and that m' is the target of m in d. We sometimes slightly abuse notation and let $t(m)$ denote a move instead of just its identifier. When $s(m)$ is an attacking (surrendering) reply to $s(m')$ we also say that m is an attacking (surrendering) reply to m'.

Protocols are in (Prakken 2005) defined as follows. A *protocol* on M is a set $P \subseteq M^{<\infty}$ satisfying the condition that whenever d is in P, so are all initial sequences that d starts with. A partial function $Pr : M^{<\infty} \longrightarrow \mathcal{P}(M)$ is derived from P as follows:

- $Pr(d) =$ undefined whenever $d \notin P$;
- $Pr(d) = \{m \mid d, m \in P\}$ otherwise.

The elements of *dom*(Pr), the domain of Pr, are called the *legal finite dialogues*. The elements of $Pr(d)$ are called the moves allowed after d. If d is a legal dialogue and $Pr(d) = \emptyset$, then d is said to be a *terminated* dialogue.

Our present **protocol** for L_c is now defined in Table 3.

For all moves m it holds that $m \in Pr(d)$ if and only if m satisfies all of the following rules:

- R$_1$: $pl(m) \in T(d)$;
- R$_2$: If $d \neq d_0$ and $m \neq m_1$, then $s(m)$ is a reply to $s(t(m))$ according to L_c.
- R$_3$: If m replies to m' then $pl(m) \neq pl(m')$.
- R$_4$: If there is an m' in d such that $t(m) = t(m')$ then $s(m) \neq s(m')$.
- R$_5$: For any $m' \in d$ that surrenders to $t(m)$, m is not an attacking counterpart of m'.
- R$_6$: If $d = \emptyset$ then $s(m)$ is of the form *claim* φ or *argue* A.
- R$_7$: If m concedes the conclusion of an argument moved in m' then m' does not reply to a *why* move.

Table 3: The protocol for L_c.

R_1 says that the player of a move must be to move. (T returns for each dialogue the player(s) to move.) R_2-R_4 formalise the idea of a dialogue as a move-reply structure that allows for alternative replies. R_5 says that once a move is surrendered, it may not be attacked any more. R_6 says that each dialogue begins with a claim or an argument; the initial claim or the conclusion of the initial argument is the *topic* of the dialogue. Finally, R_7 ensures that statements, when conceded, are conceded as claims or premises instead of as conclusions of arguments whenever possible.

The **commitment rules** of the protocol define the effects of a move on the players' commitment sets. As is well known, the agents' commitments should be carefully distinguished from their beliefs: commitments are an agent's publicly declared or accepted points of view, known to the other players, while beliefs are only known to the agent holding them; these beliefs may well be inconsistent with the agent's commitments. Below $C_{pl}(d)$ denotes the commitment set of player pl in dialogue d. At the beginning of the dialogue, the commitment sets of both players are empty and then they are updated according to the following rules.

- If $s(m) = \mathit{claim}\ \varphi$ then $C_{pl}(d,m) = C_{pl}(d) \cup \varphi$
- If $s(m) = \mathit{why}\ \varphi$ then $C_{pl}(d,m) = C_{pl}(d)$
- If $s(m) = \mathit{concede}\ \varphi$ then $C_{pl}(d,m) = C_{pl}(d) \cup \varphi$
- If $s(m) = \mathit{retract}\ \varphi$ then $C_{pl}(d,m) = C_{pl}(d) \setminus \varphi$
- If $s(m) = \mathit{argue}\ A$ then $C_{pl}(d,m) = C_{pl}(d) \cup \mathit{prem}(A) \cup \{\mathit{conc}(A)\}$

The **turntaking rule** is very liberal: the proponent starts with making a single move, then the turn switches to the opponent and after her first move it is both player's turn. Thus at any time after the second move both players can make an utterance, except that they cannot speak at the same time. In (Prakken 2005) several more strict turntaking rules are also defined and in (Bodenstaff 2005) one of them is formalised, viz. the turntaking rule for so-called relevant dialogues: in such dialogues the turn switches to the hearer after the speaker has succeeded in making the 'current state' of the dialogue favour his position (see also below).

Finally, **termination** was above implicitly defined as the situation where the player(s) to move cannot make a legal move. This means that to impose some desired termination condition (e.g. that the opponent has become committed to the dialogue topic or the proponent is not committed to the dialogue topic any more) the protocol should be defined such that there are no legal moves after the condition is satisfied.

The Event Calculus

The Event Calculus (EC) is a theory specified in first-order logic about events and their effects on states-of-affair in the world (called 'fluents' in EC). EC was originally developed by Kowalski & Sergot (1986); in this paper we use a variant of Shanahan (1999) called the 'Full Event Calculus'. Its axioms express principles like 'If an event happens at time T that initiates a some fluent then that fluent starts to hold at T' and 'If a fluent holds at T and nothing terminates it then it also holds at $T+1$'. The latter is commonly called the 'law of inertia'; it can be overruled by axioms expressing when a fluent is terminated. In applications of EC its general axioms must be supplemented with domain-specific axioms.

Our use of EC for the specification of dialogue protocols is motivated by the fact that a dialogue game can be seen as a dynamic system where dialogue utterances are events that initiate and terminate various aspects of the 'dialogical world', such as a player being the player-to-move or not, a player being committed to a certain proposition or not and a move being legal or not. Such aspects will be modelled as fluents, the value of which can change as an effect of utterances made during the dialogue.

A particularly attractive feature of EC is that it can easily be implemented as a Prolog program. This allows the modelling of temporal persistence of fluents through negation-as-failure: if termination of a fluent cannot be derived, it can be assumed to persist. Thus, for instance, a proposition added to a player's commitments can be assumed to remain a commitment until this is explicitly terminated. Also, it can be elegantly modelled that the present protocol allows replies to any earlier move in the dialogue and not only to the last move. This is modelled by the fact that the legality of a reply persists until it is explicitly terminated.

To be able to reason about fluents, they are in EC *reified*. Reification means that the fluents are treated as first-class objects so that they can be used as arguments of predicates. For example, the sentence 'at time point 2 it is the turn of player *P*' can be represented as follows.

$\mathit{HoldsAt}(\mathit{Turn}(P), 2)$

Here the formula *Turn(P)* is reified as a term to allow it to be an argument of the predicate $\mathit{HoldsAt}$.

The axioms of the Full EC make use of a number of special predicates. We now describe their informal meaning and indicate how they can be used in the specification of dialogue games. The first two predicates concern the effects of an action on the value of a fluent.

Initiates(α, β, τ) means that fluent β starts to hold after event α at time τ. This formula will be used in the following ways. Firstly, it will be used to express the addition of a statement to a player's commitment set in effect of an utterance. For instance,

$\mathit{Initiates}(\mathit{move}(1, P, \mathit{claim}\ q, 0), \mathit{CS}(P, q), 1)$

says that proponent's claim of q in his first move adds q to his commitments (the move identifier following the speech act is the move's target, in this case the dummy value 0 to express that the claim is the dialogue's first move). In a similar way it can be expressed that in effect of an utterance another move becomes legal. For instance,

$$\begin{array}{l}\mathit{Initiates}(\\ \quad \mathit{move}(1, P, \mathit{claim}\ q, 0),\\ \quad \mathit{Legal}(\mathit{move}(id, O, \mathit{why}\ q, id), t))\end{array}$$

says that a claiming of q in the proponent's first move initiates the legality of a challenge of q by the opponent. Finally, the fact that a move makes a player the player-to-move can be expressed in a similar way.

Terminates(α, β, τ) means that fluent β ceases to hold after event α at time τ. This predicate is the 'mirror predicate' of *Initiates*: it can be used in an analogous way as that predicate for expressing the deletion of a commitment, the termination of legality of a move and the termination of a player being the one to move.

At the beginning of a dialogue certain fluents will hold and certain fluents will not hold. The following two predicates can be used to express the begin situation of a dialogue.

Initially$_P(\beta)$ means that fluent β holds at the beginning of the dialogue. This formula will be used for defining the commitment set of the players and the legal moves at the beginning of the dialogue.

Initially$_N(\beta)$ means that fluent β does not hold at the beginning of the dialogue. The formula will be used to define which moves are not legal at the beginning of the dialogue, which player is not allowed to make a move and which propositions a player is not committed to.

HoldsAt(β, τ*)* means that fluent β holds at time point τ and is used to express that a fluent holds at a certain time point. For instance, the formula

$HoldsAt(Legal(move(4, O, why\ q, 1), 4))$

expresses that a challenge of q by the opponent is legal at move 4 as a reply to move 1. Such formulas will (often with variables) be used in the conditions of the rules for move legality, turntaking and termination.

Happens(α, τ_1, τ_2*)* means that event α starts at time point τ_1 and ends at time point τ_2. This predicate will be used to express all moves made during the dialogue. Since a dialogue move is assumed to have no duration, it will always hold that $\tau_1 = \tau_2$.

Besides persistence of a fluent, it must be possible to express *termination* and *initiation* of fluents at certain time points. This can be done with the predicates *Clipped* and *Declipped*; they are used in the general axioms of EC but we will not use them in our domain-specific axioms. *Clipped(*$\tau 1, \beta, \tau_2$*)* means that fluent β is terminated between times τ_1 and τ_2. *Declipped(*τ_1, β, τ_2*)* means that fluent β is initiated between times τ_1 and τ_2.

We next list the general axioms of the full EC. Following convention, variables are assumed to be implicitly universally quantified. The unique-name axioms, which are part of EC, are left implicit, as well as the usual definitions of (in)equality. The first two axioms concern the conditions that should be met in order for a fluent to *persist*. The third axiom concerns the conditions for a fluent to *terminate to hold*. Then three axioms are presented which express exactly the opposite. The fourth axiom expresses when a fluent does *not persist* and the fifth and sixth axiom express what conditions should be met for a fluent to *start to hold*. The last axiom ensures that an event takes a non-negative amount of time. Note that in our Prolog implementation the occurrences of classical negation $\neg$ in the conditions of the axioms will be implemented as negation-as-failure, to capture the law of inertia.

1. $HoldsAt(f, t) \leftarrow Initially_P(f) \wedge \neg Clipped(0, f, t)$
   This axiom states that if a fluent initially holds and is not terminated between time point 0 and time point t then the fluent still holds at time point t.

2. $HoldsAt(f, t_3) \leftarrow Happens(a, t_1, t_2) \wedge Initiates(a, f, t_1) \wedge (t_2 < t_3) \wedge \neg Clipped(t_1, f, t_3)$
   This axiom states that if event a which initiates fluent f occurs then fluent f starts to hold until fluent f is terminated.

3. $Clipped(t_1, f, t_4) \leftrightarrow \exists a, t_2, t_3(Happens(a, t_2, t_3) \wedge (t_1 < t_3) \wedge (t_2 < t_4) \wedge Terminates(a, f, t_2))$
   This axiom states that if and only if there exists an event a which occurs and terminates fluent f then fluent f is said to be *clipped*.

4. $\neg HoldsAt(f, t) \leftarrow Initially_N(f) \wedge \neg Declipped(0, f, t)$
   This axiom states that if a fluent did not initially hold and was not initiated between time point 0 and time point t then the fluent does not hold at time point t.

5. $\neg HoldsAt(f, t_3) \leftarrow Happens(a, t_1, t_2) \wedge Terminates(a, f, t_1) \wedge (t_2 < t_3) \wedge \neg Declipped(t_1, f, t_3)$
   This axiom states that if event a which terminates fluent f occurs then fluent f does not hold as long as fluent f is not initiated.

6. $Declipped(t_1, f, t_4) \leftrightarrow \exists a, t_2, t_3(Happens(a, t_2, t_3) \wedge (t_1 < t_3) \wedge (t_2 < t_4) \wedge Initiates(a, f, t_2))$
   This axiom states that if and only if there exists an event a which occurs and initiates fluent f then fluent f is said to be declipped.

7. $Happens(a, t_1, t_2) \rightarrow (t_1 \leq t_2)$
   This axiom ensures that the time an event takes can never be negative.

Formalising the P system

This section contains the main contribution of this article: our specification of the P system in EC. The following fluents will be used in addition to those of the general EC axioms.

- $move(id, p, s, tr)$ This fluent states that this is the id^{th} move where participant p states locution s targeted at tr. Initially, id will be equal to tr but they will diverge when a participant makes an illegal move: then no other fluents are changed but time moves with one unit, so the time point is raised while the move identifier is not.
- $Legal(m)$ This fluent expresses that move m is legal, where m is a tuple $move(id, p, s, tr)$. Fluents of this form hold initially for initial *claim* and *argue* moves by the proponent, while they are initiated for moves that are a well-formed reply to a certain locution, when a locution of that type is moved. So every move, when legal, initiates the legality of one or more moves as a reply to that move.
- $CS(p, \varphi)$ This fluent represents that participant p is committed to proposition φ. CS stands for commitment set; fluents of this form are initiated and terminated according to the commitment rules of the dialogue system.
- $Turn(p)$ is a fluent which is initiated when the turn shifts to participant p.
- *P* stands for proponent.
- *O* stands for opponent.
- *p* and $\bar{p}$ are defined as: $\bar{p} = O$ if and only if $p = P$ and $\bar{p} = P$ if and only if $p = O$.

Table 4 indicates to which parts of the dialogue system the various predicates pertain.

A **dialogue** is specified as a sequence of unconditional *Happens* clauses. Accordingly, the predicate $Happens$ is

Predicates	BB
$Happens(move(id,p,s,tr),t)$	Comm. Language
$Initially_P(Legal(\varphi))$, $Initially_N(Legal(\varphi))$	Protocol
$Initiates(move(id,p,s,tr), Legal(\varphi),t)$, $Terminates(move(id,p,s,tr), Legal(\varphi),t)$	Protocol
$Initially_N(CS(p,\varphi))$	Commit. Rules
$Initiates(move(id,p,s,tr), CS(p,\varphi),t)$, $Terminates(move(id,p,s,tr), CS(p,\varphi),t)$	Commit. Rules
$Initially_P(Turn(p))$, $Initially_N(Turn(p))$	Turntaking
$Initiates(move(id,p,s,tr), Turn(p_2),t)$, $Terminates(move(id,p,s,tr), Turn(p_2),t)$	Turntaking

Table 4: Predicates used

to be instantiated with a fluent $move(id,p,s,tr)$. For example, $Happens(1,P,claim(q),0)$ expresses that the proponent claimed q in the first move.

The formalisation of the **protocol** first specifies which initial moves are legal. Initially the only legal moves are a *claim* or an *argue* move by the proponent. After his first move the legality of these moves terminates.

$$\begin{array}{l} Initially_P(Legal(move(1,P,s,0))) \quad \leftarrow \\ \quad s = claim\ \varphi \vee s = argue\ A \end{array}$$

$$\begin{array}{ll} Initially_N(Legal(move(id,p,s,tr))) & \leftarrow \\ \quad p = O & \vee \\ \quad (id \neq 1) & \vee \\ \quad (s \neq claim\ \varphi \wedge s \neq argue\ A) & \vee \\ \quad (t \neq 0) & \end{array}$$

$$\begin{array}{l} Terminates(move(1,P,s_1,0), Legal(move(1,P,s_2,0)),t) \\ \quad \leftarrow \\ \quad HoldsAt(Legal(move(1,P,s_1,0)),t) \end{array}$$

The next formulas specify how the legality of non-initial moves is initiated, capturing rules R$_2$, R$_3$, R$_4$ and R$_7$ of the protocol. $Defeats(B,A)$ means 'argument B defeats argument A'. Note that the rules for *argue* moves assume that the well-formedness of arguments and their defeat relations are determined by external means.

The general format of the legality-initiating rules is as follows:

m_1 initiates the legality of m_2 if
m_1 was moved legally, and
$pl(m_2)$ is the player-to-move, and
$s(m_2)$ is a well-formed reply to $s(m_1)$, and
the specific conditions for m_2, if any, are satisfied.

Actually, in the rule for replies to initial moves only the third condition needs to be stated:

$$\begin{array}{l} Initiates(move(1,P,claim\quad \varphi,0), Legal(move(id,O,s,1)),t) \\ \quad \leftarrow \\ \quad s = why\ \varphi \vee s = concede\ \varphi \end{array}$$

The first rule for replies to non-initial moves also uses the first and second condition, while the second such rule uses all four conditions:

$$\begin{array}{ll} Initiates(move(id,p,why\ \varphi,tr), Legal(move(id_2,\bar{p},s,id)),t) & \\ \quad \leftarrow & \\ \quad HoldsAt(Legal(move(id,p,why\ \varphi,tr)),t) & \wedge \\ \quad HoldsAt(Turn(p),t) & \wedge \\ \quad ((s = argue\ A \wedge conc(A) = \varphi) \vee (s = retract\ \varphi)) & \end{array}$$

$$\begin{array}{ll} Initiates(move(id,p,argue\ A,tr), Legal(move(id_2,\bar{p},s,id)),t) & \\ \quad \leftarrow & \\ \quad HoldsAt(Legal(move(id,p,argue\ A,tr)),t) & \wedge \\ \quad HoldsAt(Turn(p),t) & \wedge \\ \quad ((s = why\ \varphi \wedge \varphi \in prem(A)) & \\ \quad \vee & \\ \quad (s = argue\ B \wedge Defeats(B,A)) & \\ \quad \vee & \\ \quad (s = concede\ \varphi \wedge \varphi = conc(A) \wedge & \\ \qquad \neg(Happens(move(tr,\bar{p},why\ \varphi,tr_2),t_2) \wedge t_2 < t) & \\ \quad \vee & \\ \quad (s = concede\ \varphi \wedge \varphi \in prem(A))) & \end{array}$$

Next the conditions are specified under which the legality of non-initial moves terminates. The first rule below says that after a move with a specific content is made, it terminates to hold as a legal move with that specific content and the same target.

$$\begin{array}{ll} Terminates(move(id,p,s,tr), Legal(move(id_2,p,s,tr)),t) & \\ \quad \leftarrow & \\ \quad HoldsAt(Turn(p),t) & \wedge \\ \quad HoldsAt(Legal(move(id,p,s,tr)),t) & \end{array}$$

The second rule captures protocol rule R$_5$ and states that when a move is a surrendering move to a target the attacking counterpart of this move at the same target terminates to be legal.

$$\begin{array}{ll} Terminates(move(id,p,s_1,tr), Legal(move(id_2,p,s_2,tr)),t) & \\ \quad \leftarrow & \\ \quad HoldsAt(Legal(move(id,p,s_1,tr)),t) & \wedge \\ \quad HoldsAt(Legal(move(id_2,p,s_2,tr)),t) & \wedge \\ \quad HoldsAt(Turn(p),t) & \wedge \\ \quad ((s_1 = concede\ \varphi \wedge s_2 = why\ \varphi) & \\ \quad \vee & \\ \quad (s_1 = concede\ \varphi \wedge s_2 = argue\ A \wedge \neg\varphi = conc(A)) & \\ \quad \vee & \\ \quad (s_1 = retract\ \varphi \wedge s_2 = argue\ A \wedge \varphi = conc(A))) & \end{array}$$

Summarising, EC's 'law of inertia' is used in this formalisation as follows. Initially, only a *claim* and *argue* move are legal and all other moves are illegal. After a move is made its legality is terminated (but only with that specific content) and the legality of well-formed replies to that move's speech act is initiated. Such legality persists until the move is made. The illegality of moves persists until its legality is initiated as just described.

As for the **commitment rules**, at the start of the dialogue the commitment sets of both players are empty. When a move is made, the speaker's commitments are updated according to the commitment rules of the P system.

$$Initially_N(CS(p,\varphi))$$

$$\begin{array}{ll} \mathit{Initiates}(\mathit{move}(id,p,s,tr), CS(p,\varphi),t) & \leftarrow \\ \quad \mathit{HoldsAt}(\mathit{Legal}(\mathit{move}(id,p,s,tr)),t) & \wedge \\ \quad \mathit{HoldsAt}(\mathit{Turn}(p),t) & \wedge \\ \quad (s = \mathit{claim}\ \varphi \vee s = \mathit{concede}\ \varphi) & \end{array}$$

$$\begin{array}{ll} \mathit{Terminates}(\mathit{move}(id,p,\mathit{retract}\ \varphi,tr), CS(p,\varphi),t) & \leftarrow \\ \quad \mathit{HoldsAt}(\mathit{Legal}(\mathit{move}(id,p,\mathit{retract}\ \varphi,tr)),t) & \wedge \\ \quad \mathit{HoldsAt}(\mathit{Turn}(p),t) & \end{array}$$

$$\begin{array}{ll} \mathit{Initiates}(\mathit{move}(id,p,\mathit{argue}\ A,tr), CS(p,\varphi),t) & \leftarrow \\ \quad \mathit{HoldsAt}(\mathit{Legal}(\mathit{move}(id,p,\mathit{argue}\ A,tr)),t) & \wedge \\ \quad \mathit{HoldsAt}(\mathit{Turn}(p),t) & \wedge \\ \quad (\varphi = \mathit{conc}(A) \vee \varphi = \mathit{prem}(A)) & \end{array}$$

The **turntaking** rules are that the first move is always made by the proponent after which the turn switches to the opponent. After the second move, the turn of the proponent is initiated but that of the opponent does not terminate.

$$\mathit{Initially}_P(\mathit{Turn}(P))$$

$$\mathit{Initially}_N(\mathit{Turn}(O))$$

$$\begin{array}{ll} \mathit{Terminates}(\mathit{move}(1,P,s,0), \mathit{Turn}(P),t) & \leftarrow \\ \quad \mathit{HoldsAt}(\mathit{Legal}(\mathit{move}(1,P,s,0)),t) & \end{array}$$

$$\begin{array}{ll} \mathit{Initiates}(\mathit{move}(id,p_1,s,tr), \mathit{Turn}(p_2),t) & \leftarrow \\ \quad \mathit{HoldsAt}(\mathit{Legal}(\mathit{move}(id,p_1,s,tr)),t) & \wedge \\ \quad ((id = 1 \wedge p_1 = P \wedge tr = 0 \wedge t = 1 \wedge p_2 = O) & \\ \quad \vee & \\ \quad (id = 2 \wedge p_1 = O \wedge tr = 1 \wedge t = 2 \wedge p_2 = P)) & \end{array}$$

A Prolog implementation

We next briefly describe a Prolog implementation of our formalisation. The source code is available at www.ewi.utwente.nl/~bodenstaffl. The implementation computes in any state of a dialogue what are the player's commitments, whether the moves made were legal, who is to move and what are the legal next moves. It can thus be used as a 'dialogue consultant' by a player, referee or external observer. A Prolog program consists of a database of clauses, which is essentially a set of if-then rules 'head if body' where the head is a first-order atom and the body a disjunction of conjunctions of literals. Facts can be specified with clauses with empty body. Variables in Prolog are written as capitals and anonymous variables are written as underscores. Constants are denoted by lowercase letters or numbers. A typical clause is of the form `A :- B,C` which means 'if B and C then A'. Logical 'or' is written as a semicolon.

In Prolog, negation is interpreted as 'negation as failure' which means that a negated atom holds if the atom itself cannot be derived. Since our EC formalisation contains classical negations, some care is needed in the transformation to Prolog clauses. The negations in the conditions of the general EC axioms can safely be translated as negation-as-failure to capture the law of inertia. Furthermore, the negations in the conditions of our specific axioms can be translated as negation-as-failure since in the current application the closed-world assumption can safely be made: a dialogue state can be completely specified, so what is not specified can be assumed false. However, for classical negations in the consequents of axioms a special predicate `not_holds_at` has to be introduced and the program has to be designed such that for no fluent both `holds_at(F,T)` and `not_holds_at(F,T)` can be derived (in our program this was straightforward). The implementation of, for example, Axiom 4 now is as follows.

```
not_holds_at(F,T) :-
  initially_n(F),
  \+ declipped(0,F,T).
```

(where `\+` denotes negation as failure). When consulting a program through the Prolog interpreter, queries can be entered. A query is, for example, `?- initiates(move(1,p,claimq,0),turn(p),2).` Prolog will try to match facts in the database with the query. If this attempt fails Prolog will try to find rules where the conclusion matches the query. Prolog will assign the values of the query to the variables in the rule . Next Prolog will try to satisfy all variables in the premises. Besides queries where Prolog is only able to answer 'yes' or 'no' it is also possible to leave anonymous variables in the query. Now Prolog will return a proper assignment to the variables if there is one. After returning the first set of possible values, the user of the interpreter can enter a semicolon after which Prolog will give another assignment if there is one, else Prolog will return 'no'.

An argument in the implementation is represented as `since(p,s)` and should be read as '`p since s`'. The support for proposition `p`, namely `s`, should be entered as a list. In Prolog a list is of the form: `[p,q,r,...]`. In our test runs no defeat relations where used but they could be added as `Initially_p` clauses. Note that, just as our above formalisation, our implementation assumes that arguments are created and defeat relations are established by some reasoner external to and combined with the Prolog program.

The following queries are useful for planning the first move.

```
holds_at(turn(P),0).
holds_at(cs(P,A),0).
holds_at(legal(move(1,P,S,0)),0).
```

In return to these queries Prolog will show whose turn it is, what the commitments of the participants are and which moves are legal at time point 0. When the dialogue proceeds these queries can be used with different time points to plan the following move and to check the effects of moves on the players' commitments. These queries can also be entered with constants and in negated form, as in `not_holds_at(legal(move(3,o,claim q,2)),4)`. To enter a move the `assert` command is used. `assert` is a built-in predicate which adds its argument as a fact to the program. Asserting a clause always succeeds in Prolog.

We now demonstrate the program by simulating the following dialogue according to the P system. The knowledge base of the proponent is $\{z; q; z, q \rightarrow a\}$ and the one of the opponent is $\{d; d \rightarrow c\}$.

Time point	Move	CS_P	CS_O
T_1	$move(1, P, claim\ a, 0)$	$\{a\}$	
T_2	$move(2, O, why\ a, 1)$	$\{a\}$	
T_3	$move(3, P, argue\ A, 2)$ $conc(A) = a$, $prem(A) = z, q$	$\{a, z, q\}$	
T_4	$move(4, O, argue\ B, 3)$ $conc(B) = c$, $prem(B) = d$	$\{a, z, q\}$	$\{c, d\}$
T_5	$move(5, P, why\ d, 4)$	$\{a, z, q\}$	$\{c, d\}$
T_6	$move(6, O, retract\ d, 5)$	$\{a, z, q\}$	$\{c, d\}$

After loading the program the fist move by the proponent is entered.

```
?− assert(happens(move(1, p, claim(a), 0), 1)).
Yes
?−
```

To plan the next move, a query is entered to find out which participant is allowed to utter which locution.

```
?− holds_at(legal(move(2, P, S, 1)), 2).
P = o
S = why(a);
P = o
S = concede(a);
No
?−
```

Only the opponent is allowed to make a move and its only legal locutions are `why a` and `concede a`. We proceed by asserting the next move. After that also the remaining part of the dialogue is given as it is simulated in Prolog.

```
?− holds_at(legal(move(3, P, A, 2)), 3).
P = p
A = argue(since(a, _G511));
P = p
A = retract_(a);
No
?− assert(happens(move(3, p, argue(since(a, [z, q])), 2), 3)).
Yes
?− holds_at(legal(move(4, P, A, 3)), 4).
P = o
A = why(z);
P = o
A = why(q);
P = o
A = argue(_G481);
No
?− assert(happens(move(4, o, argue(since(c, [d])), 3), 4)).
Yes
?− holds_at(legal(move(5, P, S, 4)), 5).
P = p
S = concede(d);
P = p
S = why(d);
P = p
S = argue(_G481);
P = p
S = concede(c);
No
?− assert(happens(move(5, p, why(d), 4), 5)).
Yes
?− holds_at(legal(move(6, P, S, 5)), 6).
P = o
S = argue(since(d, _G660));
P = o
S = retract_(d);
No
?− assert(happens(move(6, o, retract_(d), 5), 6)).
Yes
?− holds_at(legal(move(7, P, A, N)), 7).
P = o
A = concede(a)
N = 1;
P = p
A = retract_(a)
N = 2;
P = o
A = why(z)
N = 3;
P = o
A = why(q)
N = 3;
P = o
A = concede(z)
N = 3;
P = o
A = concede(q)
N = 3;
P = p
A = argue(_G500)
N = 4;
P = p
A = concede(c)
N = 4;
P = p
A = concede(d)
N = 4;
No
?−
```

It should be noted that two other moves are legal at time point 7 but are not returned by Prolog in the last question `holds_at(legal(move(7, P, A, N)), 7)`. These moves are `move(7, p, argue(since(a, Phi)), 2)` where `Phi` cannot be `[z, q]` and `move(7, o, argue(Psi), 3)` where `Psi` cannot be `[d]` because the moves with these propositions already happened at time point 3 and 4 respectively. Prolog fails to return these two clauses because not *all* available propositions are legal to use in these *argue* moves. However, when a specific query is entered, for example, `holds_at(legal(move(7, p, argue(since(a, b)), 2)), 7)`, Prolog will answer affirmatively.

Other case studies

To test the generality of the above approach we applied it in (Bodenstaff 2005) to two other protocols. Firstly, we extended the above formalisation of the P system to a stricter instantiation of the framework of (Prakken 2005), in which all moves have to satisfy a condition of relevance. This notion is defined in terms of a notion of dialogical status of a move. Briefly, a move is *in* if it is surrendered or else all its attacking replies are *out*, and a move is *out* if it has an attacking reply that is *in*. A move is *relevant* if changing the dialogical status of its target also changes the dialogical status of the initial move; the turn shifts as soon as the dialogical status of the initial move has changed. In consequence, each turn now consists of zero or more surrenders followed by zero or one attacker (if zero, then the dialogue terminates automatically). We straightforwardly added this to our formalisation of P by adding definitions of dialogical status and relevance, changing the turntaking rule and giving all rules that initiate move legality a single extra condition that the move is relevant.

Secondly, we applied our approach to the system for persuasion dialogue of Parsons, Wooldridge, & Amgoud (2003). The communication language of the PWA system was displayed above in Table 1 (which renames some terms of PWA for ease of comparison). The main differences with the P system are the absence of a locution for retractions and the replacement of the *argue* locution with a *claim* S locution for a set S of propositions; the latter is to be moved in reply to a *why* φ move. The explicit reply structure of the P system is implicitly built into PWA's protocol, where in addition a claim of a proposition can also be answered with a claim of its negation. PWA's protocol rules refer to the players' internal states, in requiring that their claims and concessions must respect their "assertion and acceptance attitudes". For instance, an agent with a "sceptical" assertion attitude may claim a proposition only if he can construct a justified argument for it on the basis of his own knowledge. As for the structural aspects of a dialogue, the PWA protocol is much stricter than that of P: essentially, in PWA the turn shifts after every move and no alternative replies to a move are allowed, except to a *claim* S move, of which each member of S can be challenged or conceded in turn. PWA's commitment rules are essentially the same as those of P. A dialogue terminates when a player is to move but can make no legal moves; in that case he has to "concede the game".

In formalising PWA in EC our above approach turned out to be generally applicable. Most of our domain-specific EC predicates turned out to be useful but some new predicates had to be added to deal with assertion and concession attitudes and with the stricter structural nature of the protocol. These features also required several new axioms and modification of existing axioms. Nevertheless, our second case study provides support for the generality of our approach.

Related work

As for related work, three publications are particularly relevant to our investigations. Firstly, Yolum & Singh (2004) apply the event calculus to reasoning about commitments to action in trading scenarios. However, they do not address argumentation and do not model reasoning about legality of dialogue moves.

Brewka (2001) reconstructs and then formalises an argumentation protocol of Rescher (1977) in a version of the situation calculus (Reiter 2001). The Rescher/Brewka system is in some respects simpler and in other respects more complex than the P system. Its underlying logic is default logic and its communication language has no explicit reply structure. Arguments are implicitly moved as claim replies to challenges. Players may make any new claim and may retract their own commitments and challenge or concede the other player's commitments at any time. The system has no turntaking rules. Dialogues terminate by convention, after which a determiner is allowed to declare one of the players the winner in a way constrained by the players' final commitments. Since Brewka distinguishes 'possible' from 'legal' moves, players can move illegal moves but then the other player can reply with an *object* locution, after which the effects of the illegal move are undone. Brewka's work was a source of inspiration for our investigations but because of the differences in dialogue systems and formalisation languages it is difficult to give a precise comparison, especially since EC and situation calculus are rather different in style.

Perhaps the closest to the present investigations is the work of Artikis, Sergot, & Pitt (2003), who formalise a modified variant of the Rescher/Brewka system in the $C+$ language of Giunchiglia *et al.* (2004) and then implement it in Giunchiglia *et al.*'s "causal calculator". The $C+$ language is closer to EC than the situation calculus (in fact, Artikis, Pitt, & Sergot (2002) used EC to formalise the contract net protocol). Artikis, Sergot, & Pitt refine Brewka's notion of legal dialogue moves into a distinction between 'permitted' and 'valid' moves. A move is valid if its player has the "power" to move it, i.e., if uttering certain words 'counts as' a certain speech act. As is well-known from legal theory, powers and permissions are logically unrelated; our 'legality' of moves corresponds to Artikis, Sergot, & Pitt's validity of moves but our approach could incorporate their refinements if required by the formalised dialogue system. The same holds for their formalisations of *object* moves and the determiner's role.

Conclusion

In this paper we investigated the suitability of the Event Calculus for formalising dialogue systems for argumentation and subsequent implementation in declarative programming languages. We have contributed to previous work in this direction as follows. Generally speaking, by applying EC to systems with some new features, our case studies have confirmed and reinforced the earlier findings that EC (and similar formalisms) are suitable for formalising dialogue systems for argumentation. More specifically, we have made the following contributions. Firstly, while in the dialogue systems studied by Brewka (2001) and Artikis, Sergot, & Pitt (2003) the protocol is mainly defined in terms of the players' commitments, we have focussed on systems in which it is largely defined in terms of an explicit reply structure of the communication language, with a distinction between attacking and surrendering replies. Secondly, we have focussed on systems with a more liberal dialogue structure and with varying turntaking rules. Finally, we have (in more detail in (Bodenstaff 2005)) shown how the constraining of dialogues by a notion of relevance can be formalised.

Our Prolog implementation of the P system illustrated that the logical formalisation of dialogue systems supports their declarative implementation. Such implementation in turn supports the design of flexible dialogue systems, in which variations in the communication language or protocol can be handled easier than when they are hard-coded in a lower-level programming language. Designing such flexible dialogue systems is one topic for future research. Another such topic is the design of intelligent agents who interact within a persuasion protocol. Agents could use a declarative implementation of a dialogue system to reason about the moves they are allowed to make and their effects, and use some internal decision-making or planning mechanism to choose from the available moves.

Acknowledgements

The research reported in this paper was partially supported by the EU under IST-FP6-002307 (ASPIC).

References

Amgoud, L.; Maudet, N.; and Parsons, S. 2000. Modelling dialogues using argumentation. In *Proceedings of the Fourth International Conference on MultiAgent Systems*, 31–38.

Artikis, A.; Pitt, J.; and Sergot, M. 2002. Animated specifications of computational societies. In *Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems (AAMAS-02)*, 1053–1062.

Artikis, A.; Sergot, M.; and Pitt, J. 2003. An executable specification of an argumentation protocol. In *ICAIL '03: Proceedings of the Ninth International Conference on Artificial Intelligence and Law*, 1–11. New York, NY, USA: ACM Press.

Atkinson, K.; Bench-Capon, T.; and McBurney, P. 2005. A dialogue game protocol for multi-agent argument over proposals for action. *Journal of Autonomous Agents and Multi-Agent Systems* 11:153–171.

Bodenstaff, L. 2005. Formalisation of argumentation protocols in event calculus. Master's thesis, Utrecht University. http://www.ewi.utwente.nl/˜bodenstaffl.

Brewka, G. 2001. Dynamic argument systems: A formal model of argumentation processes based on situation calculus. *Journal of Logic and Computation* 11(2):257–282.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence* 77(2):321–357.

Giunchiglia, E.; Lee, J.; Lifschitz, V.; McCain, N.; and Turner, H. 2004. Nonmonotonic causal theories. *Artificial Intelligence* 153:49–104.

Gordon, T. 1994. The Pleadings Game: an exercise in computational dialectics. *Artificial Intelligence and Law* 2:239–292.

Kowalski, R., and Sergot, M. 1986. A logic-based calculus of events. *New Generation Computing* 4(1):67–95.

Parsons, S.; Wooldridge, M.; and Amgoud, L. 2003. Properties and complexity of some formal inter-agent dialogues. *Journal of Logic and Computation* 13. 347-376.

Prakken, H. 2005. Coherence and flexibility in dialogue games for argumentation. *Journal of Logic and Computation* 15:1009–1040.

Rahwan, I.; Ramchurn, S.; Jennings, N.; McBurney, P.; Parsons, S.; and Sonenberg, L. 2003. Argumentation-based negotiation. *The Knowledge Engineering Review* 18:343–375.

Reiter, R. 2001. *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems*. Cambridge, MA, USA: MIT Press.

Rescher, N. 1977. *Dialectics: A Controversy-Oriented Approach to the Theory of Knowledge*. Albany, NY, USA: State University of New York Press.

Shanahan, M. 1999. The event calculus explained. *Lecture Notes in Computer Science* 1600:409–430.

Yolum, P., and Singh, M. 2004. Reasoning about commitments in the Event Calculus: an approach for specifying and executing protocols. *Annals of Mathematics and Artificial Intelligence* 42:227–253.